# Commande FIN

## 1 But

Cette commande termine proprement l'exécution d'un calcul . De cette manière, on indique au gestionnaire de mémoire que les objets peuvent être libérés et écrits sur disque dans la base globale (et permettre une POURSUITE ).

L'appel à cette procédure est **obligatoire** pour que l'exécution se termine correctement, ce qui nécessite de terminer tout fichier de commande par :

```
FIN()
```



Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

# 2 Syntaxe

```
FIN    (
          ◊  RETASSAGE =      /   'OUI',
                              /   'NON',              [DEFAUT]

          ◊  INFO_RESU =      /   'OUI',              [DEFAUT]
                              /   'NON'

          ◊  PROC0     =      /   'OUI',              [DEFAUT]
                              /   'NON'

         )
```

Un argument `exit` est disponible pour les développeurs (voir ci-après)



Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

# 3 Opérandes

La commande FIN sauvegarde l'ensemble des concepts calculés lors de l'exécution dans le fichier glob.1 (et éventuellement glob.2, ... si nécessaire), ainsi que l'ensemble des objets python du contexte d'exécution dans des fichiers pickle. Ces objets seront disponibles pour une POURSUITE du calcul.

**Remarque**

*Ne sont pas sauvegardés dans le pick.codeaster.objects, les objets python de type classe, fonction et type.*

Quand FIN a terminé son travail, Python continue à analyser les instructions qui suivent et qui vont certainement échouer si elles font appel aux objets code_aster.
Le développeur peut utiliser FIN(exit=True) pour forcer à quitter l'interpréteur Python.

## 3.1 Opérande RETASSAGE

◊ RETASSAGE = /'OUI',
/'NON', [DEFAUT]

Provoque le retassage de la base 'GLOBALE' avant écriture sur le fichier associé. Ceci permet de conserver des bases plus petites (débarrassées des objets associés aux concepts détruits par l'utilisateur).

Ce retassage est effectué comme suit par la commande :

- fermeture des bases,
- ouverture de la base 'GLOBALE',
- ouverture d'une base 'VOLATILE',
- recopie, enregistrement non vide par enregistrement non vide de la base 'GLOBALE' sur la base 'VOLATILE',
- renommage par le code de cette base 'VOLATILE' pour sauvegarde comme si c'était la base 'GLOBALE' classique.
-

La base GLOBALE est constituée d'un ou plusieurs fichiers binaires organisés sous forme d'enregistrements de longueur fixe. Lors des opérations de destruction, les enregistrements associés sont déclarés libres et peuvent éventuellement être réutilisés en cours d'exécution. En fin d'exécution, il peut subsister des enregistrements inoccupés qui contribuent à la taille finale du fichier. L'opération de retassage consiste donc à réutiliser cette place en réorganisant les enregistrements. C'est une opération qui peut entraîner de nombreuses lectures et écritures sur disque.
Lors d'une exécution en parallèle (MPI), c'est chaque instance de la base GLOBALE qui est traitée, ce qui multiplie les entrées/sorties.

## 3.2 Opérande INFO_RESU

◊ INFO_RESU = /'OUI', [DEFAUT]
/'NON',

Provoque l'impression dans le fichier MESSAGE des informations relatives au contenu de l'ensemble des structures de données résultat stocké dans la base GLOBALE.

**Remarque :**

*L'usage de ce mot clé peut augmenter de façon conséquente la durée d'exécution de la commande FIN, il est donc conseillé de modifier la valeur par défaut lorsque l'on effectue des calculs engendrant d'importantes quantités de données par leur diversité.*



Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

### 3.3 Opérande PROC0

L'opérande PROC0 dont la valeur par défaut est 'OUI', permet de restreindre l'écriture des structures de données dans la base GLOBALE sur le processeur de rang 0. Si on lui affecte la valeur 'NON', les opérations de sauvegardes seront effectuées sur tous les processeurs. Cette opération peut avoir un coût important, il peut donc être très pénalisant de l'effectuer sur chaque processeur,.
Il peut être nécessaire d'effectuer cette sauvegarde sur tous les processeurs si on enchaîne plusieurs fichiers de commandes (comm, com1, com2, etc...) dans une même exécution.

Ce mot-clé n'a de sens que dans les calculs parallèles et, sauf cas particulier, le fonctionnement est automatique.
En mode parallèle distribué (où chaque processus ne voit qu'une partie du modèle), tous les processus font les écritures ( PROC0='NON' ).
En mode parallèle centralisé (parallélisme classique), lors de l'exécution du dernier fichier de commandes, seul le processus #0 écrit la base. Lors des exécutions intermédiaires, tous les processus font les écritures.

## 4 Exemple d'impression issue de la commande FIN

L'exemple ci-dessous est extrait du fichier MESSAGE associé au cas test TTNL02A.

```
#  ------------------------------------------------------------------------
#  COMMANDE NO : 0026           CONCEPT DE TYPE :
#  -------------                -----------------
FIN( RETASSAGE='NON',
     INFO_RESU='OUI',
     PROC0='OUI',
   )


======>

STRUCTURE DU CONCEPT TEMPE    CALCULE POUR         15 NUMEROS D'ORDRE

LISTE DES NOMS SYMBOLIQUES:
! -----------!----------------!---------------!
! NUME_ORDRE !      TEMP      !   HYDR_ELGA   !
! ---------- !----------------!---------------!
!          0 !     TEMP_R     !    HYDR_R     !
!          1 !     TEMP_R     !               !
!        ... !      ...       !               !
!          9 !     TEMP_R     !               !
!         10 !     TEMP_R     !    HYDR_R     !
!         28 !     TEMP_R     !               !
!        ... !      ...       !               !
!        118 !     TEMP_R     !               !
! ---------- !----------------!---------------!

LISTE DES NOMS DE VARIABLES D'ACCES:
                                    INST             DE TYPE  R


LISTE DES NOMS DE PARAMETRES:
! -----------!-------------!-------------!-------------!-------------!
```



Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

| NUME_ORDRE | MODELE | CHAMPMAT | CARAELEM | EXCIT |
|---|---|---|---|---|
| 0 | K8 | K8 | K8 | K24 |
| 1 | K8 | K8 | K8 | K24 |
| ... | ... | ... | ... | ... |
| 118 | K8 | K8 | K8 | K24 |



Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)